WheelLoc: Enabling Continuous Location Service on Mobile Phone for Outdoor Scenarios

He Wang*, Zhiyang Wang†, Guobin Shen‡, Fan Li‡, Song Han§ and Feng Zhao‡
*Duke University, Durham, NC 27708, USA
‡Microsoft Research Asia, Beijing, 100080, China
*he.wang.ece@duke.edu; †seanwangsk@ucla.edu; ‡{jackysh,fali,zhao}@microsoft.com; §hansong8811@gmail.com

***Abstract*—The proliferation of location-based services and applications calls for provisioning of location service as a first class system component that can return accurate location fix in short response time and is energy efficient. In this paper, we present the design, implementation and evaluation of WheelLoc – a continuous system location service for outdoor scenarios. Unlike previous localization efforts that try to directly obtain a point location fix, WheelLoc adopts an *indirect* approach: it seeks to capture a user mobility trace first and to obtain any point location by time- and speed-aware interpolation or extrapolation. WheelLoc avoids energy-expensive sensors completely and relies solely on commonly available cheap sensors such as accelerometer and magnetometer. With a set of novel techniques and the leverage of publicly available road maps and cell tower information, WheelLoc is able to meet those requirements of a first class component. Experimental results confirmed the effectiveness of WheelLoc. It can return a location estimate within 40ms with an accuracy about 40 meters, consumes only 240mW energy, and effectively strikes a better energy-accuracy tradeoff than GPS duty-cycling.**

I. Introduction

Location service deserves to be a first class citizen of modern mobile systems, as evidenced by the proliferation of location-based services and applications. Location is also a key factor of increasingly popular context-aware applications including personal diaries, geofencing and location-based reminders, where continuous location awareness is required.

To become a truly "qualified" first class system component, location service should satisfy three criteria: 1) short response time; 2) high location accuracy; and 3) energy efficient. While modern mobile phone operating systems all support location service, their current implementation (typically a simple hybrid approach for better coverage or accuracy) cannot meet these criteria, as all the component localization techniques have certain drawbacks that are indispensable from their respective merits. Specifically, GPS suffers from low energy efficiency, long startup delay and canyon effect; WiFi-based and cellular-based localization methods suffer from coverage and low location accuracy, respectively. Due to the energy-hungry wireless communication, their online version have similar (if not more) energy footprints as GPS thus further suffer from low energy efficiency.

In this paper, we present the design, implementation and evaluation of WheelLoc – a system location service for continuous localization. Unlike previous localization efforts that try to directly obtain a location estimate, WheelLoc adopts an *indirect* approach: it seeks *to capture user mobility trace first and to obtain any point location through interpolation or extrapolation*, depending on when the location query is issued. Further differing from the conventional optimization path that exploits an accuracy-energy tradeoff, WheelLoc avoids expensive sensors completely. It relies solely on common cheap sensors such as accelerometer and magnetometer, and leverages existing free resources such as publicly available road maps, and the (coarse) cell tower location database that is readily collectable via crowdsourcing (e.g., [1], [2]).

WheelLoc operates continuously in the background. When the user is detected to be in motion (via accelerometer), the velocity and direction are estimated from accelerometer and magnetometer readings. Cell tower IDs are recorded at the same time. The user mobility trace is then obtained as a sequence of estimated distances and turns. Next, the mobility trace is fixed to the most likely road segments on the map via a map matching process using a Hidden Markov Model and Viterbi decoding, where the searching area is constrained by the cell towers observed. Finally, the point location for any specific time is obtained via time- and motion-state-aware interpolation or extrapolation.

Aiming at a first class system service, WheelLoc faces several challenges, including accurate distance and direction estimation to faithfully capture the mobility trace, despite the possible disorientation between the device and the real moving direction of the user (i.e., the car or bike); robust map matching against possible errors in the estimated mobility trace and the noisy cell tower location and coverage information; and finally, the energy constraint required to support continuous localization.

To cope with these challenges, we have acquired the following technical innovations which act as the major contributions of this paper. Specifically,

- We propose effective methods that faithfully capture the mobility trace for different motion states. It directly estimates the distance and direction from within the device's frame of reference, and thus avoids reorientation to the road map's global coordinate system. It also handles possible device orientation changes during the motion.
- We propose and evaluate a new application of HMMs in which we formulate cell towers as an observable state. Our model accounts for the noisy cell tower location and coverage information in the calculation of the emission and transition probabilities.

We implemented the real system on mobile phones running Android 2.3 and performed thorough evaluation for both

978-1-4673-5946-7/13/$31.00 ©2013 IEEE

component techniques and the overall system with real world traces. Experimental results demonstrate that WheelLoc can return a location fix within 40 ms for over 99% queries; the average location accuracy is around 40 meters, slightly worse than that of GPS but far better than the state-of-the-art cellular-based localization system. WheelLoc consumes about 240mW power (about 30mW is the net consumption of WheelLoc' component techniques, and around 210mW is spent on enabling background operations, which can be potentially saved.) and is significantly more efficient than the GPS that consumes about 376mW. We also compared the energy-accuracy tradeoff between WheelLoc and GPS duty-cycling and concluded that WheelLoc strikes a better tradeoff.

II. Goal and Challenges

Goal and Solution Ideas: Our goal is to design and implement a system location service that provides instant location estimate (as compared to the long cold start times of GPS) at significantly improved accuracy (relative to state-of-the-art cellular-based localization), and is readily applicable to COTS mobile devices. The service should also be energy efficient enough to afford continuous operation.

Realizing that the need of localization arises from user's mobility, and that users usually move along roads with certain transportation modes, we come up with the basic idea: to capture user mobility trace directly through motion-state-aware continuous tracking, and to infer the point location for a specific time through time- and speed-aware interpolation or extrapolation along the resulting mobility trace. To this end and ensure energy efficiency, we may exploit the *cheap*, *passive* on-device sensors such as accelerometer and magnetometer for accurate motion-state detection and rough mobility trace capture, and leverage *free* resources – publicly available road maps and cell tower location database, to determine the actual mobility trace via map matching.

The road maps and cell tower location database can be pre-cached to the mobile device, and hence enable the device to continuously locate itself without communicating with the server. The map can be compactly represented since we only need small amounts of information (essentially the two ends' location) of all road segments. For instance, the size of road map (obtained from Open Street Map) for the dense central part of Beijing (about 130 square kilometers area) is only about 375 kB. The cellular tower location database is even smaller. According to China Mobile, there are only 460,000 GSM base stations all over China by the end of 2009, and the number of cell towers in Beijing is estimated to be only several thousands. The rough position of base stations can be readily obtained from some public sources such as PlaceLab [1] and Wigle [2].

Challenges: Many prior work has successfully exploited accelerometer on mobile phone to detect user's activities [13], [19], [24], and also many previous projects have used map matching for location fix [7], [15], [20], [22]. Although these prior work has shed light on the feasibility of our approach, we still face the following challenges:

- Faithful mobility trace capture. It is critical to have good estimation of the traveled distance and also all the turns taken to ensure correct map matching results, especially in the realistic situations that possible disorientation between the device and the car may happen in the motion.
- Robust map matching. Unlike previous work that performs map matching on accurate but sparse GPS or WiFi localization results, the problem we face is the opposite: the accuracy of location estimation from cellular is very low – a magnitude worse than WiFi case, but has full coverage. In addition, cell tower location obtained from crowdsourcing can be very *noisy*.
- Energy efficiency. Our solution inevitably incurs more computation which implies more energy consumed by CPU. It is critical to carefully profile the energy expense and optimize the system to meet the design goals.

III. WheelLoc Overview

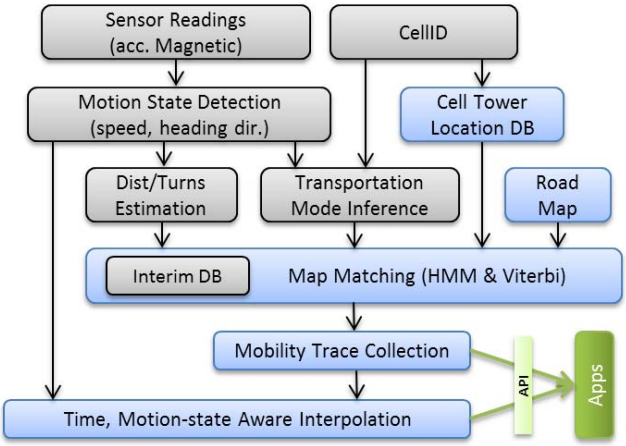

Fig. 1. WheelLoc system architecture.

The overall architecture of the proposed WheelLoc system is shown in Figure 1. It operates as follows: sensor data is sampled from the on-device accelerometer and magnetometer, and the motion state is detected together with the estimation of speed and heading direction for that state. At the same time, a very rough location fix is obtained by looking up the Cell ID in the cell tower location database cached on the device. In the following step, the distances and turns the user has traveled are estimated from the motion information; the correct road maps are selected according to transportation modes. With the selected road map, the rough location fix and also the distances and turns observed, the actual mobility trace is fixed through a map matching process using a Hidden Markov Model (HMM) and Viterbi algorithm; the cell tower location is also exploited but in a different way. Finally, the user's location at a specific time is computed through time- and motion-state-aware interpolation (or extrapolation if the user is still moving) on the specific road segment. Both the mobility trace and specific point locations can be provided to applications on demand via a set of service APIs.

In the next two sections, we will elaborate the two key component techniques of WheelLoc, namely mobility trace estimation and the map matching.

IV. Mobility Trace Estimation

Driving and cycling are of completely different motion mechanisms and thus treated differently in WheelLoc. We will

elaborate the mobility trace estimation for driving mode as it has the most significant impact on the user mobility, and briefly describe the estimation method for cycling later on.

To estimate the mobility trace (distance and direction) of the real motion of the vehicle, two challenges must be solved. Firstly, accelerometer readings are obtained from device's frame of reference ($\mathcal{M}$), whereas the real mobility trace we want to estimate is in the 3-D Global Cartesian coordinate system ($\mathcal{G}$). As the device can be arbitrarily placed, there are possible disorientation between $\mathcal{M}$ and $\mathcal{G}$. Secondly, the speed and distance estimation are very sensitive to possibly inaccurate acceleration because the integration and double integration will exaggerate the acceleration inaccuracy, especially when integrating over a long time interval.

To address the first challenge, instead of estimating the Euler angle to align $\mathcal{M}$ to $\mathcal{G}$ (as in [14]), WheelLoc directly computes the speed, distance and direction using the accelerations in $\mathcal{M}$. The correctness is shown below.

A. Heading Vector Determination

Let $\vec{a}$ and $\vec{a}'$ be two physical accelerations. Let subscription g and m indicate the respective accelerometry in $\mathcal{G}$ and $\mathcal{M}$. Assume no orientation change between the two acceleration samplings, we have $\vec{a}_m = W\vec{a}_g$ and $\vec{a}'_m = W\vec{a}'_g$, where W is the Euler transform matrix that is *unknown*. Since the W is unitary, we also have $\vec{a}_g = W'\vec{a}_m$ and $\vec{a}'_g = W'\vec{a}'_m$.

Assume the car is moving along a (possibly short) straight road segment. Then in the ideal case, the heading vector ($\vec{H}$) should be along the road segment, so does the real acceleration in $\mathcal{G}$. Because the accelerometer on most mobile phones is a gravity sensor, the readings always carry the gravity component. However, gravity is invariant, thus

$$\vec{H} = \vec{h}_g = \vec{a}'_g - \vec{a}_g$$

Now we prove that the difference of two accelerations in $\mathcal{M}$, i.e., $\vec{h}_m = \vec{a}'_m - \vec{a}_m$, is the actual embodiment of $\vec{H}$ in $\mathcal{M}$.

Theorem 1: For the device in arbitrary orientation, if there is no orientation change between the two accelerometer samplings, then the difference between the two accelerations in $\mathcal{M}$ is equivalent to the real heading vector in $\mathcal{G}$.

Proof: Let $\vec{h}^{\mathcal{G}}_m$ be the corresponding vector of $\vec{h}_m$ after converting $\mathcal{M}$ to $\mathcal{G}$ through Euler transform. Then we have

$$\vec{h}^{\mathcal{G}}_m = W'\vec{h}_m = W'W(\vec{a}'_g - \vec{a}_g)$$

As W is unitary matrix, $W'W = I$. Thus we obtain

$$\vec{h}^{\mathcal{G}}_m = \vec{h}_g = \vec{H}$$

■

Theorem 1 states that the heading vector can be obtained from device's coordinate system, as long as the orientation remains consistent. Hence, we can manipulate the acceleration in $\mathcal{M}$ directly and we will drop all the subscripts for accelerometry data from $\mathcal{M}$ hereafter.

Heading Vector Detection: The heading vector is calculated from $\vec{h} = \vec{a}' - \vec{a}$. However, the accelerometer readings are noisy. To improve the detection reliability, we first apply median filtering to the accelerometry data (sampled at 25Hz) to remove accidental spikes. Then, we calculate the mean acceleration over a running window. Here we empirically set window size to 2 second to capture relatively steady acceleration or deceleration process. Besides reducing the noise, we perform the heading vector detection only during *strong acceleration or deceleration* processes where a transient surge of acceleration is always observed. Furthermore, as $\vec{a}'$ and $\vec{a}$ have equal impact on the resulting $\vec{h}$, we perform a bi-directional search to identify better $\vec{a}'$ and $\vec{a}$.

Heading Vector Properties: We outline two properties of the heading vector that are useful for fine calibration of the detected heading vector in practice. For sake of space, we omit the proof.

Property 1: Assume no orientation change between the two accelerometer samplings, then for arbitrary motion in the horizontal plane, the heading vector in $\mathcal{M}$ is always orthogonal to the gravity in $\mathcal{M}$.

Property 2: Assume no orientation change between the two accelerometer samplings, then when turning on a horizontal plane, the heading vector in $\mathcal{M}$ is always orthogonal to the centripetal acceleration in $\mathcal{M}$.

The first property shows if we can identify chances for reliable gravity estimation such as stalls at traffic light or cruising, we can leverage these chances to fine tune the detected heading vector. The second property indicates whenever there is a turn, we should see a transient surge orthogonal to heading vector, with positive and negative surges correspond to right turns and left turns, respectively. This property can be used to fine tune the heading vector in the horizontal plane.

B. Distance and Direction Estimation

Speed and Distance Estimation: Given the heading vector $\vec{h}$, for any acceleration $\vec{a}$, we can calculate its component along the heading vector $\vec{a}_h$ as

$$\vec{a}_h = < \vec{a}, \vec{h}_{norm} > \cdot \vec{h}_{norm}$$

where $\vec{h}_{norm}$ is the normalization of $\vec{h}$. Clearly, $\vec{a}_h$ actually contributes to the vehicle's movement along the road. With known $\vec{a}_h$, it is straightforward to estimate the speed and distance through integration and double integration, respectively. However, (double) integration will exaggerate the effect of inaccurate acceleration such as the drift or initial error, especially when integrating over a long time interval.

Acceleration Error Mitigation: Mean removal is a proven effective technique [7]. It leverages the fact that, starting from a still state, the speed obtained via integration of acceleration should be zero when it comes to the next still state. In other words, any non-zero final speed must be caused by noises in acceleration. Consequently, the mean acceleration noise is removed from all accelerometer readings for the whole integration interval. WheelLoc also adopts mean removal.

In addition, WheelLoc also takes the following measures: 1) whenever strong acceleration changes are observed, the heading vector is repeatedly decided, no matter if an orientation change event is detected (see Section IV-C) or not. This helps to prevent gradual or slight orientation changes during

motion, especially after turning; 2) when long stable motion segments (e.g., still state at traffic lights or constant speed at cruising) are identified, the average accelerometer readings are taken as gravity (still in $\mathcal{M}$) and previous heading vector is re-calibrated through orthogonality. The correctness is ensured by Property 1; 3) finally, we observed that the heading vector obtained via $\vec{a}' - \vec{a}$ may vary slightly around the real heading vector. We adopt a weighted moving average process that balance the impact of history heading vectors and the current detection:

$$\vec{h}'_t = w \cdot \vec{h}_t + (1 - w) \cdot \vec{h}'_{t-1}$$

where $\vec{h}'_t$ is the determined heading vector, $\vec{h}_t$ is the newly detected heading vector and w is the weight.

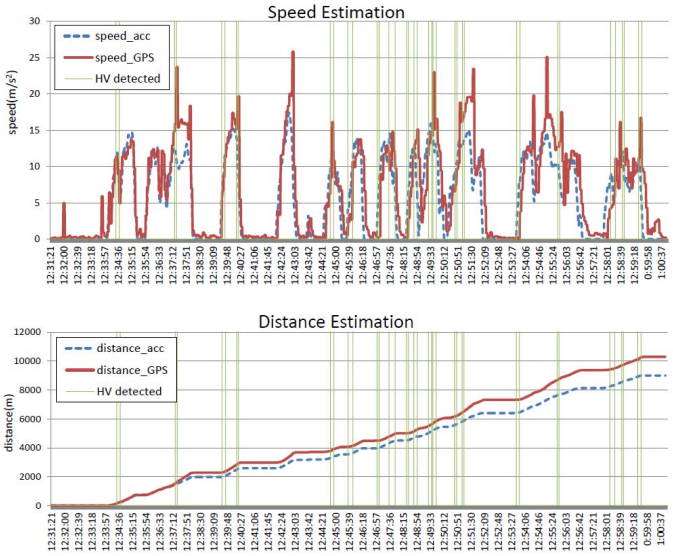

Fig. 2. Effect of acceleration error mitigation techniques. Upper: speed plot; Lower: distance plot.

Figure 2 shows the effect of the acceleration error mitigation techniques. From the figure, we can see that many heading vector detection opportunities were identified and explored. The resulting speed and distance estimation are close to those obtained using a GPS device. For this particular example, the error is within 12.6% (see Section IV-E for overall performance statistics).

Direction Estimation and Turn Detection: The vehicle's actual moving direction can be estimated by the heading vector and the orientation vector ($\vec{d}$) of the magnetic field obtained from the magnetometer in following two steps: 1) project the $\vec{d}$ to the horizontal plane and obtain the horizontal component $\vec{d}_h$:

$$\vec{d}_h = \vec{d} - < \vec{d}, \vec{a}_{v,norm} > \cdot \vec{a}_{v,norm}$$

where $\vec{a}_{v,norm}$ is the normalized vertical acceleration $\vec{a}_v$ obtained via $\vec{a}_v = \vec{a} - \vec{a}_h$; 2) calculate the angle by

$$\vartheta = \arccos(\vec{d}_h \cdot \vec{h} / |\vec{d}_h| \cdot |\vec{h}|)$$

Note that, the orientation vector $\vec{d}$ also goes through a median filtering to remove transient large orientation changes.

With the direction information, we can now determine the strong accelerometry change to be an acceleration or deceleration under the assumption of motion continuity of the vehicle, i.e., the vehicle does not make abrupt 180-degree turns.

Turns are detected by monitoring the change in the moving directions and compare against a threshold Θ. Given the instantaneous moving direction ϑ, we first obtain the stationary direction at time instant t as the mean of ϑs observed since last direction change at t_0, i.e., $\bar{\vartheta}_t = (\sum_{t_0}^{t} \vartheta)/(t - t_0)$. Then the direction change at t is computed as $\Delta\vartheta_t = \vartheta_{t+1} - \bar{\vartheta}_t$. If $|\Delta\vartheta_t| \geq \Theta$, a turning phase is spotted at t. In our implementation, we have empirically set Θ to 25 degrees.

C. Orientation Change Detection and Handling

In WheelLoc, the orientation change of the device is detected with two heuristics: 1) significant change of the vertical acceleration. This is achieved by monitoring the vertical acceleration component (i.e., $\vec{a}_v$). If the car is moving horizontally, then in an ideal case, there should be no change along the vertical direction. 2) significant change of direction in short time period. This is achieved by monitoring the angle ϑ between the heading vector and the magnetometer's north direction. If the car is moving along the straight line, ϑ should be very stable. If the car is making a real turnings, it usually needs several seconds. Thus, the duration of the change process can be used to tell orientation changes from real turnings.

Once we identify a suspicious orientation change, we mark down the time and put the device into an under determined state. If we detect a strong acceleration change, we re-estimate the heading vector, and trace back till the beginning of this orientation change. If we detect multiple consecutive suspicious orientation changes, we will discard those intermediate ones, which are most likely caused by either short user body activities or passing a bumpy road segment. We simply assume the motion state (both speed and heading direction) is the mean over the previous and subsequent steady motion states.

D. Cycling Cases

The physical motion mechanisms of cycling are completely different from that of driving. This leads to two fundamental differences that impact the readings from the accelerometer and magnetometer. Firstly, there is persistent variations in device's orientations in the cycling cases and the variation can be very large when the phone is in pant pocket. This renders the distance and direction estimation method designed for driving scenarios not applicable. Secondly, the variations are highly cyclic. This property, absent in the other scenarios, can be exploited in a design specific to cycling.

Speed and Distance Estimation: We leverage the periodicity of cycling to estimate the speed and distance. Despite that people may ride bikes differently, we found that there is always a strong correlation between the pedal frequency and the corresponding speed.[1] Figure 3 plots the measured speed (in km/h) versus the pedal frequency (cycles in last 20 seconds). We can see the obvious linear correlation. Therefore, we simply approximate the motion speed as a linear function

[1]The conclusion may become invalid if the bike is multiple speed and the user changes the speed gear frequently. However, most cases people ride bikes at a fixed transfer ratio.

of pedal frequency which can be obtained by looking into acceleration readings. The slope of the linear function depends on the bike type and also the personal riding pattern, and can be learned after a successful map matching (e.g., selecting a road segment, divide its length by the number of pedalling cycles). In WheelLoc, the slope is default to 17 km/h/Hz. In real life, many cyclist coast for considerable distance where they do not pedal. We estimate the speed of coasting as the average between the speed immediately before and after the period.

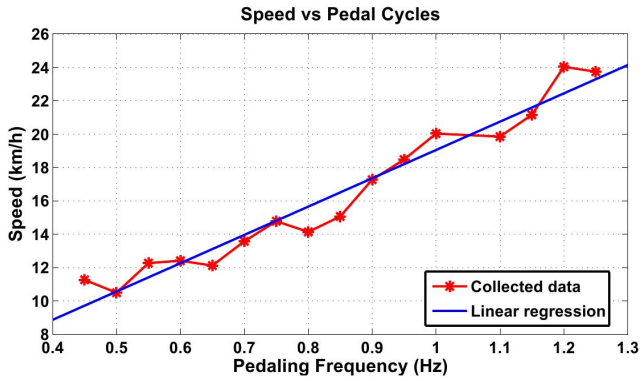

Fig. 3. Correlation between pedalling frequency and speed. Speed data are calculated from GPS readings.

Direction Estimation: The consistent changing of device orientation makes it an extremely difficult task to detect the direction of user's motion because each orientation implies a different coordinate system. To be orientation agnostic, we use the magnitude of accelerometer readings. Through extensive experiments, we found that the instantaneous direction from the magnetometer usually corresponds to the correct heading direction when the magnitude of acceleration reaches the minimum, as illustrated by the dashed vertical lines in Figure 4. Therefore, we simply take this favorable fact to our implementation and take the median of so-obtained directions in last three cycles as current motion direction. For phones in other body positions (i.e., upper body and hip pocket), the orientation of phone does not change much and the magnetometer readings are stable. Therefore, we handle all the cases in a unified way.

E. Performance Evaluation

We collected large amount of user mobility traces and separated them into driving only or cycling only traces, then performed distance/direction estimation on each trace. The results are compared against the ground truth obtained from the real map and shown in Figure 5. We see that we can achieve distance estimation within about 10% from the real distance for most of time, and the maximum difference is around 20%. For over 90% of time we obtain direction estimation within 25 degrees from the real direction. Note that, we did not impose any restriction on the device orientation during the trace collection.

V. Map Matching

With the correct road map (indicated by the motion states) and also the rough estimation of the mobility trace, we apply map matching to find the most likely sequence of actual road segments.

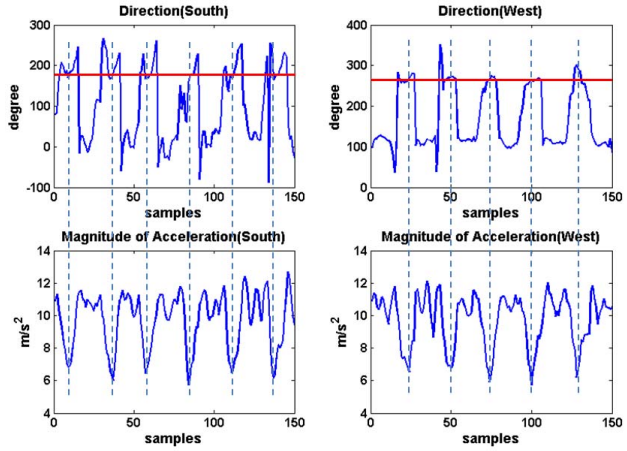

Fig. 4. Illustration of the instant to snapshot the heading direction. The horizontal solid line is the real direction. There are a few jumps in the upper plots caused by the 360 and 0 degree transition in magnetometer readings.

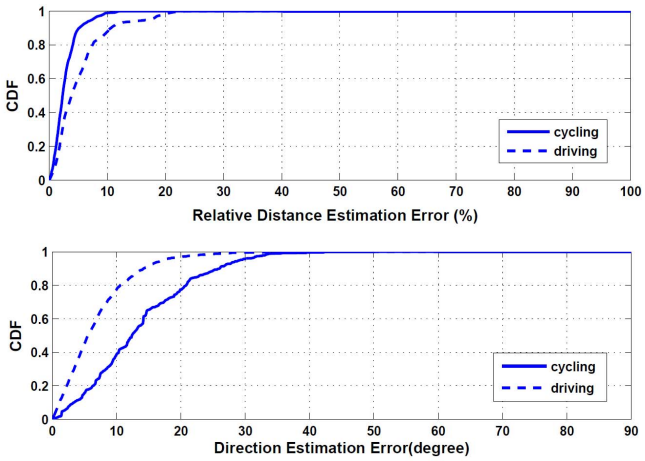

Fig. 5. Performance of distance (upper) and direction estimation (lower) for driving and cycling.

A. HMM and Viterbi Algorithm

We choose Viterbi decoding over Hidden Markov Model (HMM) as the underlying map matching algorithm, as also adopted in [7], [15], [20], [22]. Our formulation of HMM differs from those previous work in two aspects:

1) WheelLoc observes a rough mobility trace represented as a sequence of traveled distances and turns. This is unlike previous work that observes a series of sparse but relatively accurate position samples obtained from either GPS or WiFi. WheelLoc also considers the actual heading directions at turns;
2) WheelLoc also observes the cell towers which provides full coverage but very rough location estimation (order of magnitude less accurate than that from GPS or WiFi), and exploits them to hint the possible range to perform the matching;

Hidden states: The hidden states in our HMM refer to spans of road segments. A span of road segment is defined as one or more continuous road segments between one intersection and another (possibly arbitrary) intersection.

Observations: In WheelLoc, we select three types of observations, namely angle of turns, distance between two consecutive turns, and the cell tower ID from which rough locations are looked up from the cell tower location database.

Emission probabilities: An emission probability describes the probability of an observation *given* a specific hidden state. The cell tower provides a rough estimation of the position at sampling time. Since cell tower ID is regularly updated every second, we may have multiple samples along a span of road segments due to user mobility. Consequently, we can use the joint probability density $\prod N(dist(l_t, c_t))$ as the emission probability, where $N(\cdot)$ is a zero mean Gaussian function (due to noisy cell tower locations) with variance R, the radius of cell tower coverage. In WheelLoc, R is a system parameter and we will evaluate its impact later on. l_t is the uniformly interpolated position at sampling time t along the specific hidden state, c_t is the location of the cell tower observed at t, and $dist(l_t, c_t)$ is the Euclidian distance between l_t and c_t.

Transition probabilities: A transition between hidden states is signaled when a turn is detected. The transition probability is computed from all the observations at the turning point. We impose several constraints to limit the number of candidate hidden states, including bound-limiting on estimated angles of turns, error distribution of estimated distances, and cell tower coverage.The transition probability from hidden state i to hidden state j can be computed as

$$P(j|d) = \frac{P(d|j)P(j)}{\sum P(j)P(d|j)}$$

where the priori probability $P(j)$ and the conditional probability $P(d|j)$ are calculated as follows:

- $P(j) = \frac{1}{N}$, where N is the total number of candidate states. At the start state (i.e., $t = 1$), the candidates include all intersections lying within the coverage of cell tower observed at the turn except those with no outgoing states that lie within the direction range ($\gamma \pm \varepsilon_\gamma$). If $t > 1$, the candidate states are identified with the following three conditions: 1) egresses from state i, i.e., whose first segment joins the last segment of state i; 2) the angle of turn is within the direction range $\gamma \pm \varepsilon_\gamma$; and 3) the end point is within the coverage of the cell tower observed at the turn.
- $P(d|j)$, the conditional probability of an estimated distance d being at the candidate state j, is calculated as

$$P(d|j) = \frac{1}{\sigma_j\sqrt{2\pi}} e^{\frac{-(d-d_j)^2}{2{\sigma_j}^2}}$$

  where d_j is the actual length of the state j, and σ_j reflects the variance of our distance estimation algorithm when the actual length is around d_j, assume the distance estimation error follows Gaussian distribution.

Note that if we have multiple cell tower information available (e.g., a device that can hear multiple cell towers), we may further narrow down the scope of candidate states, for the calculation of both the emission and transition probabilities. In another extreme case where the cell tower location is not available, no cell tower constraint will be applied.

Viterbi algorithm: We adopt Viterbi algorithm to determine the most likely sequence of hidden states corresponding to a series of captured mobility trace based on above HMM formulation. The resulting hidden states represents a fix to the actual mobility trace, i.e., the actual road segments traversed by the user.

B. Point Location Fix

Once the real mobility trace is determined, user's location at any given time can then be obtained through interpolation or extrapolation on the resulting sequence of road segments while considering user's motion speed.

C. Performance Evaluation

Unlike previous work, our HMM formulation introduces a set of uncertain system parameters, including noisy cell tower location and coverage, and inaccurate distance and direction estimation. We study their impact via simulations that is set up based on real road map. In all the simulations, we vary one parameter at a time with all other parameters set to the default values obtained in real experiments. For instance, when evaluating the cell tower information, we set the distance and direction estimation error to be uniformly distributed within 20% and $\pm$25 degrees (i.e., from Figure 5), respectively.

Figure 6 shows the performances under various noisy cell tower information. Figure 6-(a) examines the impact of cell tower location inaccuracy under different coverage settings. We can see that our algorithm is robust to the cell tower location noise, especially when the coverage is relatively large. However, the performance drops when the location accuracy is very low and the cell coverage is also small. The reason is that small cell coverage may mistakenly exclude the right candidate states. Figure 6-(b) further reveals that when the cell tower location is relatively accurate, larger cell coverage may lead to reduced performances because it becomes less selective, i.e., its constraint strength on the emission and transition probabilities is weakened.

In practice, due to the crowdsourcing nature of our cell tower location database, we will *miss the cell tower location information when encountering new cell towers*. The impact of missing cell tower location is shown in Figure 6-(c). We can see that our algorithm is very resistent to missing cell tower location. In the worst case when the cell tower location is not available at all, our algorithm still correctly fix over 85% road segments. Interestingly, comparing Figure 6-(a) and (c), we find that wrong cell tower location is worse than not using it.

VI. Implementation and Evaluation

A. System Implementation

We have implemented WheelLoc as a continuous background system service on NexusOne phones running Android 2.3, following the architecture depicted in Figure 1. WheelLoc collects sensory data continuously and performs necessary processing (e.g., motion state detection, distance estimation and turn detection) periodically (every 4 seconds) to maintain small memory footprints. To ensure timely response, WheelLoc performs map matching whenever a turn is detected instead of waiting until a location fix request is received.

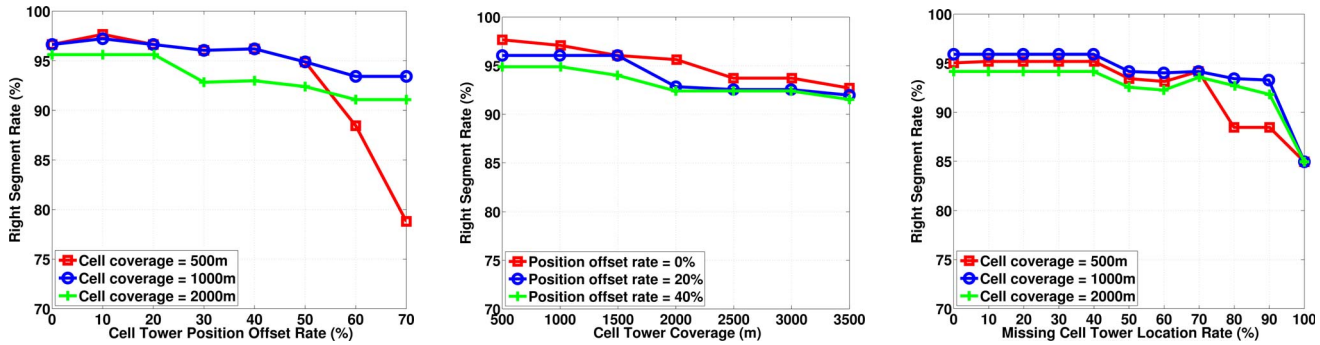

Fig. 6. Impact of the cellular network: (a) Left-most plot: inaccurate cell tower locations, (b) Plot in the middle: uncertainty of cell tower coverage, (c) Right-most plot: missing cell tower locations.

Motion State Detection: Since there exists a lot of research on state detection by sensors ([4], [9], [12], [19], [25]), we only mention one acceleration feature we found to separate driving from other states here. The key observation is that engine and road will give small (but detectable) vibrations even when the vehicle is in steady speed. The vibration frequency in detection window turns out to an effective feature to identify driving state.

Energy Optimization: We observe that people tend to have low frequency state changes in real life. Moreover, most of the time spent is in (casual) still state. WheelLoc leverages such opportunities for additional energy savings. Specifically, when the device is detected to have been in *absolute* still state for over 5 minutes, we start to duty cycle the sampling process to sample only four seconds per minute. In this situation, AlarmManager API is used to put the CPU into sleep.

B. System Evaluation

Experiment Statistics: To measure the performance of WheelLoc, we collected over 320km real driving traces in 20 data collection passes. Taxis were mostly employed and five different users were involved. Data were collected during typical traffic conditions, including normal traffic conditions, lunch time, rush hours, and mid-night extreme light traffics. The average speed ranges from 10 km/h to 80 km/h. The data collection area is in the north-west part of Beijing, about 63 km^2 in area and contains 650 road segments. The cycling traces were collected in the same area but from two other volunteers, with over 60km in length. In all these experiments, users were free to place their phones in any pocket and allowed to change the device orientations during the data collection course.

The performance of WheelLoc is measured against the three criteria of a first class system location service, that is, system response time, localization accuracy and the energy efficiency.

Response Time: The response time is measured by replaying the traces at the same pace as if the trace data were obtained from the sensors. We then randomly issue location fix requests and measure the response time. Figure 7 shows the CDF of the system response time. From the figure, we can see that over 99% of time, our system can return a location fix within 40ms. There is a noticeable trend change around the point (12ms, 81%). The extra latency is due to our implementation in which the data is locked during the periodical speed and distance estimation in the background.

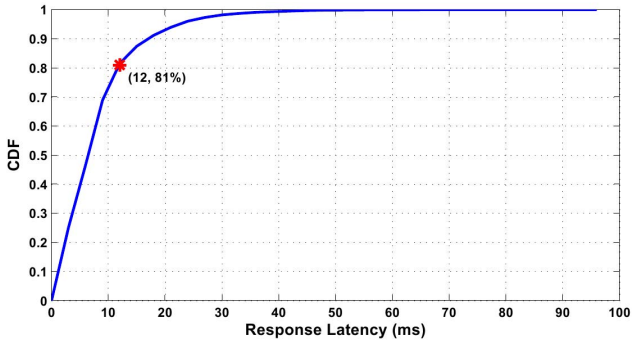

Fig. 7. The CDF plot of WheelLoc's response time.

Location Accuracy: To measure the location accuracy, the exact ground truth position is required, which is very challenging to acquire during the experiments with cars or bicycles. In our experiments, we intuitively established the ground truth positions by projecting the GPS data to the correct road segments and treat the projected position as the ground truth location. Evidently, such practice actually favors the GPS since its error is now the distance to the road segment instead of to the exact point location.

TABLE I. ACCURACY COMPARISON BETWEEN USING THE HARD-TO-OBTAIN GROUND TRUTH AND OUR PROJECTION-BASED TREATMENT OVER A 20KM DRIVING TRACE, 32 CROSSROADS MARKED.

Means	Marked		Projected	
	mean (m)	std (m)	mean (m)	std (m)
GPS	**38.7**	17.8	**11.3**	7.2
WheelLoc	35.9	27.0	25.2	19.0
Cellular	167.8	98.7	160.1	114.1

To verify this, we leveraged uniquely identifiable locations such as crossroads. We collected a 20km driving trace in which the precise times when subjects pass by crossroads (32 in total) are marked. The exact location of these crossroads are obtained from a fine road map, and their GPS locations are retrieved from the trace using those timestamps. Since WheelLoc can precisely match all the turns, we only compare the accuracy of those passed-through crossroads for fairness. The results are shown in Table I. Clearly, our way to establish the ground

truth indeed favors the accuracy of GPS: its accuracy at those marked crossroads is much worse than that calculated our way. But such treatment does not favor the accuracy of WheelLoc as much, nor to cellular-based localization.

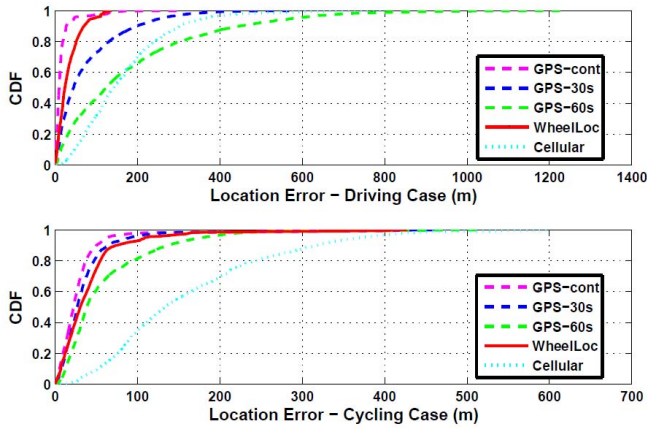

Fig. 8. Performance comparison between WheelLoc, GPS and a commercial cellular localization service.

Figure 8 shows the end-to-end localization performance of WheelLoc, and the comparison against GPS at different sampling intervals and also the state-of-the-art cellular-based localization service (GoogleMap's location service) for both driving and cycling cases. As expected, WheelLoc significantly outperforms the cellular-based localization, and is only slightly worse than continuous GPS localization. In particular, for over 82% of time, WheelLoc returns a location estimate within 50 meters to the ground truth. For the driving case, its performance is significantly better than GPS duty-cycled at 30 seconds or every minute. Interestingly, for the cycling case, its accuracy is very close to that of the GPS. The reason is that the cycling speed is much slower than driving, and the speed change of cycling tends to be steadier, hence, easier to predict.

Table II summarizes the statistics of location accuracy for all the traces we experimented. We can see that for driving cases, WheelLoc achieves mean localization error of about 32 meters, which is much smaller than that from the cellular-based service and slightly worse than that from the continuous operating GPS (about 16 meters). Note that, we have used the projected GPS location to measure its location error. The actual GPS location error can be much larger, as shown in Table I. Similar observations are also made from cycling cases, where WheelLoc can achieve mean localization error of about 41 meters. However, both GPS and WheelLoc perform worse in the cycling cases because the bike lane is closer to buildings thus impairing GPS readings.

TABLE II. MEAN AND STANDARD DEVIATION OF LOCALIZATION RESULTS FOR DRIVING AND CYCLING FOR ALL THE TRACES.

Means	Driving		Cycling	
	mean (m)	std (m)	mean (m)	std (m)
GPS	15.9	24.7	29.8	43.1
WheelLoc	**32.2**	27.9	**41.2**	47.7
Cellular	163.7	109.7	162.3	105.6

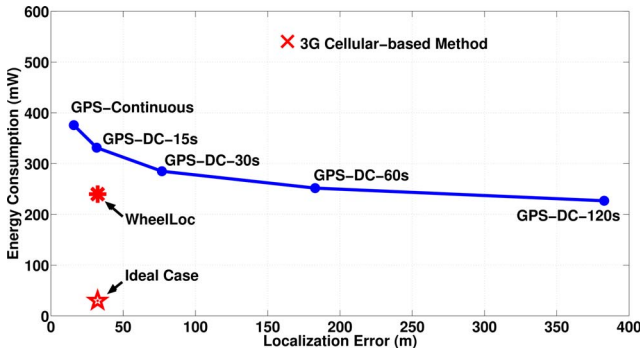

Fig. 9. Comparison of the energy-accuracy tradeoff between WheelLoc, 3G cellular-based method and GPS duty-cycling schemes.

Energy Efficiency: We profiled the energy consumption of different localization means on a NexusOne phone (running Android 2.3) and obtained the following average results: GPS consumes 424.6 mW and 376 mW when scanning for satellites and performing location fix, respectively; online cellular-based resolution consumes 720.8 mW and 540.6 mW for using GPRS and 3G, and 349.5 mW for WiFi. In comparison, the overall power consumption of WheelLoc is about 240mW.

Nowadays, most of mobile phone processors are designed to handle intensive applications and user interactions, thus they are not well suited for running continue sensing tasks in the background. For example, in our case, we have to keep processor awake by the WakeLock while WheelLoc is sampling sensors. Serval works have suggested to use additional low power processors to handle continue sensing tasks more energy-efficiently ([10], [17], [18]). Apart from consuming lower power in the active mode, another advantage of low power processors is the negligible wakeup overheads [3]. With these features of low power processor, continuous sensing tasks are much more energy efficient, which could make WheelLoc outperform GPS more significantly in terms of energy efficiency. The *net* power consumption of WheelLoc is only about 30mW (i.e., 12-fold more energy efficient than GPS) if WakeLock is hacked away.

Energy-Accuracy Tradeoff: While consuming less energy, WheelLoc is not as accurate as a GPS device that operates continuously. On the other hand, duty-cycling GPS can also reduce energy consumption at the cost of reduced accuracy. It makes more sense to compare the energy-accuracy tradeoff. We measure the GPS energy consumption at different duty cycling intervals. To obtain the location accuracy of GPS under duty-cycling, we sampled the real traces collected and interpolate the locations when GPS is off duty.

Figure 9 shows the resulting energy-accuracy tradeoff for different GPS duty-cycling settings, 3G cellular-based method, and also the tradeoff point achieved by WheelLoc. We can see that WheelLoc indeed strikes a better energy-accuracy tradeoff. We also plot the "ideal" tradeoff that can be achieved by hacking away the WakeLock. This "ideal" tradeoff significantly outperforms the GPS duty-cycling and indicates the potential of WheelLoc. Note that, in this paper, we have deliberately avoided using GPS to see how good the system may perform. In practice, there are situations where GPS may still be used

by the user. WheelLoc can take advantage of such occasional use of GPS to fix its location.

VII. Related Work

Many previous work explored the accuracy-energy tradeoff using a variety of available location sensing mechanisms. Gaonkar *et al* [6] proposed to dynamically switch between multiple localization means based on the displacement error estimation. Enloc [5] seeked to obtain the optimal localization accuracy for a given energy budget using predictions. In [8], position updates are scheduled to minimize energy consumption and optimize robustness, based on the estimation and prediction of system conditions and mobility. Paek *et al* [16] did the similar optimization and avoided GPS readings by blacklisting GPS unavailable areas. a-Loc [11] automatically assesses the dynamic accuracy requirement and fulfills it with most suitable sensors available. WheelLoc deviates significantly from the accuracy-energy tradeoff path. It seeks to achieve accurate location using only cheap sensors, and completely avoids expensive ones.

Map matching was explored in few recent work as an effective means to improve the accuracy of GPS [15] and WiFi [22] based location fixes. In [20], the authors applied map matching to track public transit where the underground route is also considered. In CTrack [21], two-pass HMM is used to obtain rough location estimates from cellular readings and sensor hints. However, CTrack only uses binary hints, namely "moving/static" and "turn/straight". Guha *et al* [7] designed special hardware to better estimate the trace and fix it to map via map matching, the device was assumed to have no orientation change during tracking. Woodman *et al* [23] also applied map-matching but in an indoor scenarios and used particle filtering. WheelLoc also adopts map matching, but differs in HMM formulation due to different hidden state definition and the incorporation of both distance/direction estimation and (possibly inaccurate) cell tower location information. WheelLoc uses COTS mobile phone and targets at real applicability where different transportation modes and possible orientation changes of the device may appear.

VIII. Conclusion

In this paper, we present WheelLoc, an energy efficient continuous location service for outdoor scenarios that completely relies on cheap on-premise sensors such as accelerometer and magnetometer. WheelLoc is able to provide instant location fix with significantly improved energy-accuracy tradeoff, as compared with GPS and cellular-based localization, respectively. WheelLoc adopts an indirect approach by first capturing the user mobility trace and obtaining point locations afterwards. It unleashes the accelerometer for accurate, position- and orientation-agnostic motion state detection, and together with the magnetometer, for faithful mobility trace capture. It also leverages the road map and the rough cell tower location information to resolve the captured mobility trace to the actual trace in the physical world through map matching and novel HMM formulation. Both simulations and real world experiments confirm the effectiveness of WheelLoc. We believe WheelLoc is a successful push towards a first class system location service.

References

[1] Place lab. http://www.placelab.org.

[2] Wigle. http://wigle.net.

[3] TI. OMAP 5 mobile processors: OMAP 5 platform, 2011.

[4] L. Bao and S. S. Intille. Activity recognition from user-annotated acceleration data. pages 1–17. Springer, 2004.

[5] I. Constandache, S. Gaonkar, M. Sayler, R. R. Choudhury, and L. Cox. Enloc: Energy-efficient localization for mobile phones. In *IEEE INFOCOM Mini Conference*, 2009.

[6] S. Gaonkar and R. R. Choudhury. Micro-blog: map-casting from mobile phones to virtual sensor maps. In *SenSys '07*, pages 401–402, 2007.

[7] S. Guha, K. Plarre, D. Lissner, S. Mitra, B. Krishna, P. Dutta, and S. Kumar. Autowitness: locating and tracking stolen property while tolerating gps and radio outages. In *SenSys '10*, pages 29–42, 2010.

[8] M. B. Kjaergaard, J. Langdal, T. Godsk, and T. Toftkjaer. Entracked: energy-efficient robust position tracking for mobile devices. In *MobiSys '09*, pages 221–234, 2009.

[9] J. Lester, T. Choudhury, and G. Borriello. A practical approach to recognizing physical activities. In *Pervasive*, pages 1–16, 2006.

[10] F. X. Lin, Z. Wang, R. LiKamWa, and L. Zhong. Reflex: Using low-power processors in smartphones without knowing them. In *ASPLOS '12*, 2012.

[11] K. Lin, A. Kansal, D. Lymberopoulos, and F. Zhao. Energy-accuracy trade-off for continuous mobile device location. In *MobiSys '10*, pages 285–298, 2010.

[12] H. Lu, J. Yang, Z. Liu, N. D. Lane, T. Choudhury, and A. T. Campbell. The jigsaw continuous sensing engine for mobile phone applications. In *SenSys '10*, 2010.

[13] D. Mizell. Using gravity to estimate accelerometer orientation. In *ISWC '03*, pages 252–, 2003.

[14] P. Mohan, V. N. Padmanabhan, and R. Ramjee. Nericell: rich monitoring of road and traffic conditions using mobile smartphones. In *SenSys '08*, pages 323–336, 2008.

[15] P. Newson and J. Krumm. Hidden markov map matching through noise and sparseness. In *GIS '09*, pages 336–343.

[16] J. Paek, J. Kim, and R. Govindan. Energy-efficient rate-adaptive gps-based positioning for smartphones. In *MobiSys '10*, pages 299–314, 2010.

[17] B. Priyantha, D. Lymberopoulos, and J. Liu. Littlerock: Enabling energy-efficient continuous sensing on mobile phones. In *IEEE Pervasive Computing '11*, 2011.

[18] M.-R. Ra, B. Priyanthay, A. Kansaly, and J. Liu. Improving energy efficiency of personal sensing applications with heterogeneous multiprocessors. In *UbiComp '12*, 2012.

[19] S. Reddy, M. Mun, J. Burke, D. Estrin, M. Hansen, and M. Srivastava. Using mobile phones to determine transportation modes. *ACM Trans. Sen. Netw.*, 6:1–27, March 2010.

[20] A. Thiagarajan, J. Biagioni, T. Gerlich, and J. Eriksson. Cooperative transit tracking using smart-phones. In *SenSys '10*, pages 85–98, 2010.

[21] A. Thiagarajan, L. Ravindranath, H. Balakrishnan, S. Madden, and L. Girod. Accurate, low-energy trajectory mapping for mobile devices. In *NSDI '11*, 2011.

[22] A. Thiagarajan, L. Ravindranath, and et al. Vtrack: accurate, energy-aware road traffic delay estimation using mobile phones. In *SenSys '09*, pages 85–98, 2009.

[23] O. Woodman and R. Harle. Pedestrian localisation for indoor environments. In *UbiComp '08*, pages 114–123, 2008.

[24] J. Yang. Toward physical activity diary: motion recognition using simple acceleration features with mobile phones. In *IMCE '09*, pages 1–10, 2009.

[25] Y. Zheng, Y. Chen, X. Xie, and W.-Y. Ma. Geolife2.0: A location-based social networking service. In *MDM '09*, pages 357–358, 2009.